Date 2001-06-20	ISO/IS 10303-24
Secretariat ANSI/NIST	ISO/TC 184/SC4 WG11 N165

Report of Voting/Annex B

MEMBER BODY	COMMENTS	OBSERVATIONS OF THE PROJECT

USA	ISSUE NUMBER: USA-1
	CLAUSE: 6.8.1
	CLASSIFICATION: Technical, minor
	
	DESCRIPTION:
	Memory Management Issues Get Attribute (Explicit)
	When a derived attribute is accessed, new, non-persistent, entity instances may be created (depending on the EXPRESS definition of the attribute). Neither Part 22 nor Part 24 specifies how these instances should be managed. Part 24 needs to specify minimal lifetime for these temporary instances to be available (the way it does for strings, binaries, and enumerations in 4.3.3).
	
	Part 203 includes several examples of such derived attributes including axis1_placement_2d.z :
	
	`ENTITY axis1_placement`
	`  SUBTYPE OF (placement);`
	`  axis : OPTIONAL direction;`
	`DERIVE`
	`  z : direction := NVL(normalise(axis),direction([0,0,1]));`
	`WHERE`
	`  WR1: SELF\geometric_representation_item.dim = 3;`
	`END_ENTITY; -- axis1_placement`
		Accepted

MEMBER BODY	COMMENTS	OBSERVATIONS OF THE PROJECT
	PROPOSED SOLUTION: Proposed Solution 1: The temporary instances will only be available until the next derived attribute is accessed. Proposed Solution 2: Add two functions to the API: sdaiDeleteDerivedValues (SdaiInstance, SdaiAttr att); sdaiDeleteDerivedValuesBN (SdaiInstance, SdaiAttr att); The temporary instances created for a specific attribute of an instance will be available until the application programmer requests that they be deleted. RESOLUTION Updated 4.3.3 to reflect the first proposed solution	
USA	ISSUE NUMBER: USA-2 CLAUSE: 6.8.1 CLASSIFICATION: Technical, minor DESCRIPTION: Memory Management Issues Get Attribute (Explicit) For inverse attributes, Part 22 (10.10.1) specifies that a NPL is created, but this then places a burden on the application programmer to know that the attribute being accessed is inverse, and then explicitly delete the NPL which was implicitly created.	**Accepted**

MEMBER BODY	COMMENTS	OBSERVATIONS OF THE PROJECT
	PROPOSED SOLUTION: Proposed Solution 1: The NPL implicitly created for an inverse attribute shall be valid (only) until the next call that evaluates an inverse attribute. Proposed Solution 2: The NPL must be explicitly deleted with sdaiDeleteNPL() by the application programmer. (This is just an editorial change) RESOLUTION Updated 4.3.3 to reflect the first proposed solution	

USA	ISSUE NUMBER: USA-3
	CLAUSE: 6.8.1, 6.10.7, 6.13.1
	CLASSIFICATION: Technical, minor
	DESCRIPTION:
	The standard is unclear when doing an sdaiGetXXX operation with an ADB parameter, if the ADB is implicitly created by the get function. Which of the following patterns is correct?
	
	Pattern 1:
	`SdaiADB adb;`
	`sdaiGetAttrBN (inst, "foo", sdaiADB, &adb);`
		Accepted

MEMBER BODY	COMMENTS	OBSERVATIONS OF THE PROJECT

	Pattern 2:
	`SdaiADB = sdaiCreateEmptyADB();`
	`sdaiGetAttrBN (inst, "foo", sdaiADB, &adb);`
	
	RESOLUTION
	Clarified the descriptions of 6.8.1, 6.8.13.1, 6.10.7 and 6.13.1 to indicate that the pattern #2 must be used.	